TP 2 – Gestion des versions, des tags en cohérence avec les exigences et la documentation

Exercice 1 – Ajout d'une documentation au projet “login-generator”

1. Github propose le dispositif Github Pages pour associer un site de documentation à un projet, une organisation ou un utilisateur. Étudiez la documentation de Github Pages (https://help.github.com/categories/github-pages-basics/) et identifiez le moyen qui vous semble le plus pertinent pour associer une documentation à votre projet “login-generator”.
2. **Après validation par votre enseignant**, initiez la documentation de votre projet.
3. Configurer votre projet pour que la documentation soit visible via l’URL https://<utilisateur ou organisation>.github.io/<nom du projet>/

Exercice 2 – Création de la version 1.0.0

1. Sur l’espace Github de votre projet, créez un “milestone” intitulé “release-1.0.0” ayant pour “due date” la date du jour.
2. Créez deux exigences sous forme d’issues Github exprimant les fonctionnalités de “login-generator” à ce stade. Affectez les deux exigences au milestone “release-1.0.0”.
3. Marquez les issues comme “closed”. Est-il possible de retrouver facilement toutes les exigences attachées à un “milestone” ? Testez.
4. Modifiez votre fichier “pom.xml” pour que la version de l’artefact corresponde bien à “1.0.0”.
5. Dans votre repository Git local, créez le tag “1.0.0”. Poussez le tag sur Github. Naviguez sur le site Github pour identifier comment se traduit le “push” d’un tag.

Exercice 3 – Vers la version 1.1.0

Dans cette exercice, l’objectif est de développer une nouvelle fonctionnalité permettant de générer un mot de passe aléatoire à la demande.

1. Créez un nouveau milestone “release-1.1.0” ayant pour “due date” la date du jour.
2. Créez les exigences suivantes sous forme d'issue github en les affectant au milestone créé précédemment :
   - “Génération d'un mot de passe aléatoire.”
   - “Documentation génération d’un mot de passe aléatoire”
3. Créez dans votre repository local la branche intitulée “password_generation” et basculez sur cette branche.
4. Modifiez le numéro de version dans le fichier “pom.xml” afin que celui-ci corresponde à la situation de développement.
5. Implémentez, testez et documentez la fonctionnalité. On pourra par exemple ajouter la classe “PasswordGeneration” fournissant la méthode “String getRandomPassword()” retournant un mot de passe de 8 caractères généré de manière aléatoire.
6. Modifiez le numéro de version dans le fichier “pom.xml” afin que celui-ci

corresponde à la situation de livraison en préparation.
7. Fusionnez votre branche master avec la branche “password_generation” en mode “no fast forward”.
8. Supprimez votre branche “password_generation”.
9. Dans votre repository Git local, créez le tag “1.1.0”. Poussez la branche master et le tag sur Github. Vérifiez que la deuxième release apparaît bien sur Github. Fermez les issues du milestone en cours si cela n’a pas été fait à l’aide des commits.

Exercice 4 – Correction d'un bug affectant les versions 1.0.0 et 1.1.0

Nous allons maintenant nous atteler à la correction d’un bug affectant les versions 1.0.0 et 1.1.0 : si le nom d’un utilisateur contient moins de 3 caractères la génération de login échoue.

1. Créez un nouveau milestone “release-1.0.1” ayant pour “due date” la date du jour. Créez une issue décrivant le bug et affectez la à ce milestone.
2. Créez la branche “hotfix” à partir du tag “1.0.0” et basculez sur cette branche.
3. Modifiez le numéro de version dans le fichier “pom.xml” afin que celui-ci corresponde à la situation de développement
4. Complétez les tests unitaires permettant de couvrir le code de la classe LoginGenerator en utilisant le cas de test suivant :
   quand on génère le login de "Paul Du", on vérifie que le login généré et ajouté à la liste des logins existants est "PDU".

Après avoir constaté que ce test ne passait pas :

5. Modifiez le code de la classe LoginGenerator afin que vos tests passent.
6. Effectuez la(les) dernière(s) modification(s) et opérations Git permettant de tagger et de publier la version 1.0.1 en liant le(s) commit(s) à l’issue correspondant au bug.
7. Effectuez la(les) dernière(s) modification(s) et opérations Git permettant de tagger et de publier la version 1.1.1 en liant le(s) commit(s) à l’issue correspondant au bug.
8. Supprimez la branche “hotfix”.
9. Dans notre organisation actuelle, est-il possible de retrouver sur Github toutes les issues qui sont apparues uniquement en version 1.1.0, les fonctionnalités présentes uniquement en 1.0.0 ou les fonctionnalités présentes en 1.0.1 mais pas en 1.1.0 ? Imaginez un dispositif tirant partie des labels permettant de simplifier de telles recherches.
10. **Après validation par votre enseignant**, mettez en œuvre ce dispositif.